

Nimble Algorithms for Cloud Computing

Ravindran Kannan *

Santosh Vempala †

Abstract

Cloud computing is a new paradigm where data is stored across multiple servers and the goal is to compute a function of all the data. We consider a simple model where each server uses polynomial time and space, but communication among servers being more expensive is ideally bounded by a polylogarithmic function of the input size. We will dub algorithms that satisfy these types of resource bounds as *nimble*.

The main contribution of the paper is to develop nimble algorithms for several areas which involve massive data and for that reason have been extensively studied in the context of Streaming Algorithms. The areas are approximation of Frequency Moments, Counting bipartite homomorphisms (number of copies of a fixed bipartite graph H in a graph G), Rank- k approximation to a matrix, and Clustering. For frequency moments, we will use a new importance sampling technique based on high powers of the frequencies. We reduce the problem of counting homomorphisms to estimating implicitly defined frequency moments. For rank- k approximations, besides recent results of several authors developed in the Streaming context, we use a variant of the random projection method. For clustering, we use our rank- k approximation and the small *coreset* of Chen [15] of size at most polynomial in the dimension.

In contrast to our algorithms in the cloud computing model, in the streaming model, known lower bound results for frequency moments and rank- k approximations rule out the existence of algorithms that use polylogarithmic space.

*Microsoft Research India. Email: kannan@microsoft.com

†Georgia Tech. Email: vempala@gatech.edu

1 Introduction

Cloud Computing is a new paradigm for storage and processing of massive data. The first objective of this paper is to formulate a clean high-level model of Cloud Computing. The bulk of the paper develops algorithms in this model. In addition to time and space, we measure communication as a critical resource for cloud computing algorithms. While there have been several models of Parallel and Distributed Computing, the Streaming model is perhaps the closest in spirit [31, 12]. Surprisingly, we find that natural problems, such as the computation of frequency moments and low-rank matrix approximation are feasible in this model while they are known to be provably infeasible in the Streaming Model.

In the frequency moment problem, the data consists of updates to the counts of elements, stored on many servers.

Problem 1.1 (Frequency Moments) *A nonnegative n -vector of frequencies of n distinct elements, $f = (f_1, f_2, \dots, f_n)$, is represented by a sequence of updates, each of the form (i, x) , which indicates “increment f_i by x ” for some $x > 0$. The objective is to compute the k ’th moment of f , namely, $\sum_{i=1}^n f_i^k$, where k is a positive integer.*

We also consider a related problem of **counting homomorphisms**, i.e., the number of copies of one (small) graph in another (large) graph.

In many applications, the data is a massive matrix has to be split across servers. One would like to compute Linear Algebra quantities of the whole matrix, but without having to communicate it across servers. Here, we study the following fundamental problem in this area. Given a matrix A stored across servers, find an approximation B to A of rank at most k . The best approximation can be found by Singular Value Decomposition as is well-known. But here, we will be satisfied with a near-optimal B (where “near-optimal” is in terms of relative error).

Problem 1.2 (Low-rank Approximation) *Given a $n \times d$ matrix A , a positive integer k and $\varepsilon > 0$, find an $n \times d$ matrix B of rank at most k such that*

$$\|A - B\|_F \leq (1 + \varepsilon) \cdot \min_{X: \text{rank}(X) \leq k} \|A - X\|_F.$$

[Here, for a matrix A , the Frobenius norm $\|A\|_F^2$ is the sum of squares of the entries of A .]

We also consider one of the most popular variants of **Clustering** based on the k -means objective function. We will develop algorithms for these problems in the cloud setting. In contrast, as we state more precisely below, in the streaming model, known lower bounds rule out such algorithms.

1.1 The Model

Our model is simple: there are s servers, where s is generally to be thought of as a constant (but may be a function of the size of the problem as well). Each server has a part of the input data of the problem. For example, if it is a graph problem, the servers might have disjoint subsets of edges of the graph. A more general example is a matrix. We consider two distinct models.

In the **row partition** setting, the rows of the matrix are partitioned and each server has a subset of (whole) rows, whereas, in the **arbitrary partition** setting (called “turnstile model” in streaming), the matrix is given by a set of updates of the form (A_{ij}, α) which says “increase A_{ij} by α ”, where, α is *positive or negative* real number. We assume that the updates are partitioned

arbitrarily among the servers. Since each server has polynomial time and space, it can just as well add up all the updates for each entry A_{ij} and thus server t has an $n \times d$ matrix A_t such that the whole matrix A is given as

$$A = A_1 + A_2 + \cdots + A_s.$$

We will measure two resources: (1) time taken by the servers to solve a problem and (2) total communication among servers. Of these, we treat communication as the more expensive resource and generally restrict it to be polylogarithmic (sometime sublinear) in the size of the problem, whereas, we will generally allow the time taken by each processor (as well as the internal Random Access Memory) used by each processor to be polynomially bounded in the data size. To complete the model description, we will say that a problem **has a nimble algorithm** if there is a randomized algorithm which solves the problem with initial data partitioned among the s processors arbitrarily, using polynomial time and polynomial space in each server and sublinear (ideally polylogarithmic when $s \in \tilde{O}(1)$) amount of communication. We will generally state explicit bounds on the resources used by each algorithm.

Such a distributed model was introduced by Cormode et al [18], and subsequently studied by others, including Philips et al [32] and Woodruff and Zhang [34]. The latter give algorithms and lower bounds for estimating frequency moments, we mention their results presently. This model has also been recently considered for distributed learning problems [7, 26, 24]. As observed in [25], it is no weaker than streaming in the following sense: Any sketching algorithm (i.e., one that can be applied to arbitrary subsets of data and combine their outputs in arbitrary order) that uses $O(s)$ space and p passes over the data can be implemented in the cloud with $O(sp)$ communication and the same asymptotic time complexity.

1.2 Results

Our first result is a nimble algorithm for estimating frequency moments.

Theorem 1.1 *For any positive integer k , the k 'th frequency moment of data presented as updates partitioned arbitrarily among s servers can be estimated to within relative error ε with probability at least $3/4$ using $O((2s)^k \log n / \varepsilon^2)$ communication and $O(n)$ time per server.*

Prior to our work, Woodruff and Zhang [34] gave an algorithm that achieves $s^{k-1} \left(\frac{C \log n}{\varepsilon} \right)^{O(k)}$ communication.

We can contrast the above result with the streaming model. Alon, Matias and Szegedy's seminal paper showed that frequency moments for $k \leq 2$ can be computed in polylog space, and polynomial space for k greater than a larger constant. This was improved to nearly matching bounds all k , with an upper bound of $\tilde{O}(n^{1-2/k})$ [27] and the lower bound showing that the k 'th frequency moment for $k > 2$ needs $\Omega(n^{1-2/k} / \log n)$ memory [3, 8, 13].

The main idea of our nimble algorithm is to sample elements from within a server according to higher moments. It turns out that sampling according to the squared value, which has been very effective for other settings, does not suffice here. There is also a nearly matching lower bound which is a direct consequence of the communication complexity of the multi-party set disjointness problem [3, 13, 8].

Theorem 1.2 *Estimating the k 'th frequency moment of a set to within a factor of $(1 + \varepsilon)$ in the cloud model with s servers needs $\Omega(s^{k-1} / \varepsilon \log k)$ communication.*

We next turn to another counting problem, namely counting homomorphisms, i.e., the number of copies of a small graph H in a large graph G , when the vertices of H (rows of its adjacency matrix) are partitioned arbitrarily among servers. This is a natural problem with many interesting special cases, such as counting the number of k -cycles, stars/cliques of a fixed size etc. We will show that for a large class of graphs, the number of homomorphisms can be estimated to relative error by a nimble algorithm, assuming the vertices are partitioned arbitrarily among servers (i.e., the row partition model). We state here the result for counting complete bipartite graphs (which includes the case of stars $K_{1,t}$ and 4-cycles).

Theorem 1.3 *The number of complete bipartite subgraphs $K_{r,t}$ in a given graph $G = (V, E)$ can be estimated to relative error $(1 + \varepsilon)$ by a nimble algorithm.*

As we will see in Section 3, we can in fact count the number of bipartite subgraphs, in which each vertex on one side must have its degree belonging to some given set of integers, e.g., the number of bipartite subgraphs $K_{r,t}$ with degree on the left at least $t/2$ (i.e., the degrees are constrained to the set $S = \{\lceil t/2 \rceil, \lceil t/2 \rceil + 1, \dots, t\}$). We note that we cannot approximately count the number of cliques of size r (even triangles) with polylogarithmic communication; this is perhaps not surprising in the light of nearly linear lower bounds in the streaming model [31, 12].

Our next set of results are for low-rank approximation. We begin with the row partition model¹.

Theorem 1.4 *Suppose the rows of the input $n \times d$ matrix A are partitioned among s servers arbitrarily with an $n_t \times d$ matrix A_t in server t . For any $1 \geq \varepsilon > 0$, there is a nimble algorithm that, on termination, leaves an $n_t \times d$ matrix C_t in server t such that the matrix C formed by all the C_t 's achieves*

$$\|A - C\|_F \leq (3 + \varepsilon) \min_{X: \text{rank}(X) \leq k} \|A - X\|_F$$

using linear space, polynomial time and with total communication bounded by $O(sk/\varepsilon)$ rows of A and $O(sk^2/\varepsilon^2)$ additional real numbers.

At the heart of the algorithm is a procedure to approximate the top k singular vectors with low communication. In Section 4, we first develop simpler algorithms with somewhat higher communication bounds of $O(sd^2)$ to solve the problem exactly (Theorem 4.2), and $O(skd)$ to get a factor 3 approximation (Theorem 4.3) before indicating a proof of Theorem 1.4. Note that if $s, k, d \in \tilde{O}(1)$, then the communication is polylogarithmic. The guarantee above is stronger for matrices whose rows are sparse.

Our next result is for the arbitrary partition model.

Theorem 1.5 *Consider the arbitrary partition model where an $n \times d$ matrix A_t resides in server t and the data matrix $A = A_1 + A_2 + \dots + A_s$. For any $1 \geq \varepsilon > 0$, there is a nimble algorithm that, on termination, leaves a $n \times d$ matrix C_t in server t such that the matrix $C = C_1 + C_2 + \dots + C_s$ with high probability achieves*

$$\|A - C\|_F \leq (1 + \varepsilon) \min_{X: \text{rank}(X) \leq k} \|A - X\|_F$$

using linear space, polynomial time and with total communication complexity $\tilde{O}(sd^2/\varepsilon^2)$.

¹The hidden constants in our Ω, O notation are all independent of n, k, d .

This result uses a $r \times n$ psuedo-random projection matrix P . Alon, Matias and Szegedy [3] use P with full independence among rows, but with only $O(1)$ -way independence within a row to save space; they show for one fixed vector $x \in \mathbf{R}^n$, $\|Px\|^2$ estimates $\|x\|^2$ to relative error. Here, we will use P with $\tilde{O}(d)$ -way independence within a row and full independence among rows with $r \in \tilde{O}(d)$. We will prove that with high probability, simultaneously for all $x \in R^d$, we have $\|PAx\|^2$ estimates $\|Ax\|^2$.

Finally, we consider the problem of Clustering multi-dimensional data. This problem has received much attention in the traditional computational settings (polynomial-time approximation algorithms [23, 14, 6, 30, 15, 17]). More to the point here, it is an important problem for many modern large data sets and has therefore been considered extensively in the setting of streaming algorithms as well more recently in parallel and distributed machine learning settings (see e.g., Chapters 3,4,5 in [9]).

We now define the clustering problem precisely. Data points are rows of a $n \times d$ matrix A . The rows of A are partitioned among s processors with the t 'th processor having n_t rows which form an $n_t \times d$ matrix A_t . A k -clustering is defined by an $n \times d$ matrix C of centers, where, each row of C is one of the k centers and the i 'th row of C is the cluster center closest to the i th data point, namely, the i th row of A .

One approach to clustering is to use a stochastic model of data, say a mixture of k Gaussians, with the assumption that the means of the Gaussians are well-separated so that the clusters are distinct and then devise algorithms to find the clusters. The separation assumed could be roughly stated as “the means of any two different Gaussians are at least $\Omega_k(1)$ standard deviations apart”. For general Gaussians, the variance could be different in different directions; so one takes the maximum variance in any direction.

We do not assume any stochastic model of data. Following [30, 17], we can define an analog of standard deviation: for a clustering C , define $\sigma(C)$ by

$$\sigma(C)^2 = \frac{1}{n} \max_{\|v\|=1} \|(A - C)v\|^2.$$

In words, $\sigma(C)^2$ is the maximum average squared distance of any data point to its center along a direction v , the maximum taken over all directions v . Clearly, $\sigma(C)$ is just $1/\sqrt{n}$ times the spectral norm of $A - C$.

Definition 1.1 *A clustering C is said to be proper if every pair of centers of C is at least $c_0(k)\sigma(C)$ apart (where, $c_0(k)$ is a function of k alone).*

Definition 1.2 *Any k -clustering C' with the property that to every center μ of C , there is a center ν of C' with $|\mu - \nu| \leq c_1(k)\sigma(C)$, is said to be a valid approximation to C .*

The reason for this definition is that if there is a proper clustering C and we find a valid approximation C' to C , then it is not difficult to show that C and C' differ only in a small fraction of the data points (with a suitable choice of c_0, c_1). Can such a valid clustering be computed by a nimble algorithm? Surprisingly, the answer is yes, in the row partition model.

Theorem 1.6 *Suppose the rows of an $n \times d$ matrix A are the data points to be k -clustered and the rows of A are partitioned among s servers. Assume that there is a proper k -clustering C . There is a nimble algorithm which finds a valid approximation to C . The algorithm takes polynomial time and uses $O(d^2 + k^4)$ total communication for $s = O(1)$ servers.*

2 Estimating frequency moments

Let f_{ij} denote the frequency of the i 'th element in the j 'th server (i.e., the sum of all updates to f_i on the j th server) when there are n distinct elements and s servers. Then the k 'th frequency moment of the data is

$$\sum_{i=1}^n \left(\sum_{j=1}^s f_{ij} \right)^k.$$

A fundamental problem is to estimate estimate frequency moments efficiently.

2.1 Two servers

To warm up, we consider the problem of estimating the third moment when data is split between two servers. Let u_i, v_i denote the frequencies of the i 'th element in the two servers, so that $f_i = u_i + v_i$. Then we can use the following algorithm:

1. The two servers independently compute u_i^3 and v_i^3 .
2. The first server samples an i.i.d. subset S according to the distribution that samples j with probability

$$p_j = \frac{u_j^3}{\sum_{i=1}^n u_i^3}$$

and announces their frequencies in its data subset. The second server computes $A = 3 \sum_{j \in S} u_j^2 v_j / p_j$ and announces.

3. The servers reverse their roles and estimate $B = 3 \sum_{j \in S} u_j v_j^2 / q_j$ where now the sample S is drawn by the second server, according to q proportional to v_j^3 .
4. The final estimate is $A + B + \sum_i u_i^3 + v_i^3$.

Lemma 2.1 *Let X be a random variable set to $u_j^2 v_j$ with probability $p_j = u_j^3 / \sum_{i=1}^n u_i^3$. Then,*

$$\mathbb{E}(X) = \sum_{i=1}^n u_i^2 v_i \quad \text{and} \quad \text{Var}(X) \leq \left(\sum_{i=1}^n u_i^3 + v_i^3 \right)^2.$$

Proof. We bound the variance by the second moment. Let $J = \{i : u_i, v_i > 0\}$.

$$\begin{aligned} \text{Var}(X) &\leq \sum_{i \in J} \frac{(u_i^2 v_i)^2}{p_i} \\ &= \left(\sum_{i=1}^n u_i^3 \right) \left(\sum_{i \in J} u_i v_i^2 \right) \\ &\leq \left(\sum_{i=1}^n u_i^3 \right) \left(\sum_{i=1}^n u_i^3 + v_i^3 \right) \\ &\leq \left(\sum_{i=1}^n u_i^3 + v_i^3 \right)^2. \end{aligned}$$

□

From the lemma it follows that we can estimate the sum of the two mixed terms in the expansion of $(u_i + v_i)^3$ to within ε times the third moment using $O(1/\varepsilon^2)$ samples. Thus, the third moment of element frequencies can be estimated in the cloud with two servers to within $1 + \varepsilon$ relative error with probability at least $3/4$ using $O(\log n/\varepsilon^2)$ communication and $O(n)$ time.

Next we estimate the k 'th moment by extending the above algorithm.

1. The two servers separately compute u_i^k and v_i^k .
2. The first server samples an i.i.d. subset S according to the distribution that samples j with probability

$$p_j = \frac{u_j^k}{\sum_{i=1}^n u_i^k}$$

and announces the frequencies of the sampled elements in its data subset. The second server computes

$$A = \sum_{r=\lceil k/2 \rceil}^{k-1} \binom{k}{r} \sum_{j \in S} \frac{u_j^r v_j^{k-r}}{p_j}$$

and announces.

3. The servers reverse their roles and estimate

$$B = \sum_{r=\lceil k/2 \rceil}^{k-1} \binom{k}{r} \sum_{j \in S} \frac{u_j^{k-r} v_j^r}{q_j}$$

where now the sample S is drawn by the second server, according to q proportional to v_j^k .

4. The final estimate is $A + B + \sum_i u_i^k + v_i^k$.

Lemma 2.2 *Let $r \geq k/2$ and X be a random variable equal to $u_j^r v_j^{k-r}/p_j$ with probability $p_j = u_j^k / \sum_{i=1}^n u_i^k$. Then*

$$\mathbb{E}(X) = \sum_{j=1}^n u_j^r v_j^{k-r} \quad \text{and} \quad \text{Var}(X) \leq \left(\sum_{i=1}^n (u_i^k + v_i^k) \right)^2$$

Proof. The proof is a direct extension of the case $k = 3$. Let J be the subset of indices where u_i, v_i are both nonzero.

$$\begin{aligned}
\text{Var}(X) &\leq \sum_{i \in J} \frac{(u_i^r v_i^{k-r})^2}{p_i} \\
&= \left(\sum_{i=1}^n u_i^k \right) \left(\sum_{i \in J} u_i^{2r-k} v_i^{2k-2r} \right) \\
&\leq \left(\sum_{i=1}^n u_i^k \right) \left(\sum_{i=1}^n u_i^k + v_i^k \right) \\
&\leq \left(\sum_{i=1}^n u_i^k + v_i^k \right)^2.
\end{aligned}$$

We used the fact that for any $r \geq k/2$,

$$u_i^{2r-k} v_i^{2k-2r} \leq \max\{u_i^k, v_i^k\}.$$

□

Theorem 2.3 *For two servers and any constant k , the k 'th moment of element frequencies can be estimated in the cloud with two servers to within $1 + \varepsilon$ relative error with probability at least $3/4$ using $O(\log n/\varepsilon^2)$ communication and $O(n)$ time.*

2.2 The general setting

Here we consider the setting of $s \geq 2$ servers. First we note that there is a lower bound of $\Omega(s^{k-1})$ from the communication complexity of the multi-party set disjointness problem defined as follows: given a universe of n elements and t players receiving a subset, solve the promise problem: Distinguish between the subsets being disjoint OR the subsets having exactly one element in common to all of them and no other intersection between any of them.

Theorem 2.4 [13, 8] *The communication complexity of multi-party set disjointness with n elements and t players is $\Omega(n/t \log t)$.*

This theorem readily implies the lower bound of Theorem 1.2, by the reduction of [3].

Proof. (of Theorem 1.2.) The reduction from set disjointness to k 'th moment estimation is essentially the same as the one given in [3]. Each of the s servers gets m distinct elements with frequency 1. Moreover, either the subsets of elements for each server are completely disjoint or there is one element common to all s servers. Thus the frequency moment is either sm or $s(m-1) + s^k$. Estimating the frequency moment to a factor less than $(s(m-1) + s^k)/sm$ solves the set disjointness problem and thus requires $\Omega(sm/s \log s) = \Omega(m/s \log s)$ communication. Setting $m = 2s^{k-1}/\varepsilon$ proves the corollary. □

We next describe the nimble algorithm that nearly matches this lower bound.

Recall that the frequency of the i 'th element in the j 'th server is f_{ij} . We expand the k 'th moment:

$$\begin{aligned} \sum_{i=1}^n \left(\sum_{j=1}^s f_{ij} \right)^k &= \sum_{i=1}^n \sum_{\sum_{j=1}^s r_j = k} \binom{k}{r_1, \dots, r_s} \prod_{j=1}^s f_{ij}^{r_j} \\ &= \sum_{\sum_{j=1}^s r_j = k} \binom{k}{r_1, \dots, r_s} \sum_{i=1}^n \prod_{j=1}^s f_{ij}^{r_j}. \end{aligned}$$

We estimate each internal sum separately as follows. Consider a generic term

$$\sum_{i=1}^n f_{i1}^{r_1} \cdots f_{im}^{r_m} \tag{1}$$

where $r_1, \dots, r_m \geq 1$ and $\sum_{j=1}^m r_j = k$. Let $M = \max_{i,j} f_{ij}$.

1. Each server j classifies its frequencies f_{ij} into at most $\log M$ buckets, with the l 'th bucket B_l having indices i with $f_{ij} \in [2^{l-1}, 2^l]$.
2. Each server j does the following:
 - (a) For each bucket l , server j samples a set S of indices from the bucket according to $p_i = f_{ij}^k / \sum_{t \in B_l} f_{tj}^k$ and announces them.
 - (b) Every other server announces frequencies of indices in S and have $f_{ij} \leq 2^l$. Let $I = \{i \in S : f_{ij} \leq 2^l \text{ for } j = 2, 3, \dots, s\}$.
 - (c) Server j computes

$$\frac{|B_l|}{|S|} \cdot \sum_{i \in I} \prod_{j=1}^m f_{ij}^{r_j}.$$

3. The estimate for (1) is the sum of computations of each server for each bucket.

Lemma 2.5 *Define the random variable X as*

$$X = \frac{\prod_{j=1}^m f_{ij}^{r_j}}{p_i} \chi(i \in I) \text{ with prob. } p_i = \frac{f_{i1}^k}{\sum_{t \in B_l} f_{t1}^k} \text{ for } i \in B_l.$$

Then

$$\mathbb{E}(X) = \sum_{i \in I} \prod_{j=1}^m f_{ij}^{r_j} \quad \text{and} \quad \text{Var}(X) \leq 2^{2k} \left(\sum_{i=1}^n f_{i1}^k \right)^2.$$

Proof. To bound the variance (of a single random sample), we use the second moment. If $2r_1 \geq k$, then the proof is very similar to that of Lemma 2.2. Assume $2r_1 < k$.

$$\begin{aligned}
\text{Var}(X) &\leq \sum_{i \in I} \frac{\prod_{j=1}^m f_{ij}^{2r_j}}{p_i} \\
&= \left(\sum_{i \in B_l} f_{i1}^k \right) \sum_{i \in I} \frac{\prod_{j=2}^m f_{ij}^{2r_j}}{f_{i1}^{k-2r_1}} \\
&\leq \left(\sum_{i \in B_l} f_{i1}^k \right) \sum_{i \in I} \frac{\prod_{j=2}^m (2f_{i1})^{2r_j}}{f_{i1}^{k-2r_1}} \\
&= 2^{2k} \left(\sum_{i \in B_l} f_{i1}^k \right)^2.
\end{aligned}$$

□

This lemma readily implies Theorem 1.1.

3 Counting homomorphisms

We consider the problem of counting the number of copies of a fixed graph H in a larger graph G , when the vertices of G are partitioned arbitrarily among cloud servers.

The simplest example is counting the number of paths of length 2. This can be written as

$$t(K_{1,2}, G) = \sum_{i=1}^n \binom{d_i}{2}$$

where d_i is the degree of vertex i . Expanding, we get

$$t(K_{1,2}, G) = \frac{1}{2} \sum_{i=1}^n d_i^2 - \sum_{i=1}^n d_i.$$

Both terms are frequency moments where degree d_i is the “frequency” of the i ’th vertex. So we can use the frequency moment estimation algorithm of the previous section. (Since these are only first and second moments so far, we could also do it in the streaming model, with vertices arriving in arbitrary order, using the approach of [3]).

Next suppose we want to count the number of stars with t leaves, i.e., $K_{1,t}$. This is

$$t(K_{1,t}, G) = \sum_{i=1}^n \binom{d_i}{t}$$

which is a polynomial in the first t frequency moments.

To count the number of 4-cycles, let us define d_{ij} to be the number of common neighbors of vertices i and j . Then,

$$t(C_4, G) = \sum_{i \neq j} \binom{d_{ij}}{2}$$

This is again just the first two moments of the “frequencies” d_{ij} . We now prove its generalization to any complete bipartite subgraph.

Proof.(of Theorem 1.3.) To count the number of complete bipartite subgraphs $K_{r,t}$, we define the joint degree d_S of a subset S of r vertices as the number of neighbors common to all r vertices in S . Then

$$t(K_{r,t}, G) = \sum_{S \subset V, |S|=r} \binom{d_S}{t}$$

This needs only the first t frequency moments of these set degrees. Since vertices are partitioned in the cloud, we can keep track of these subset vertex degrees and use the frequency moment estimation algorithm from the previous section. \square

This can be generalized to counting all bipartite subgraphs that satisfy degree constraints of a certain type.

Theorem 3.1 *Let S be an arbitrary subset of $[t]$. Let \mathcal{H} consist of all bipartite graphs $H = (U, V, E)$ with $r = |U|$, $t = |V|$ s.t. for every $u \in U$, $\deg(v) \in S$. Then the total number of copies of elements of \mathcal{H} that occur as subgraphs of a graph G , $t(\mathcal{H}, G)$ can be estimated to relative accuracy ε by a nimble algorithm using $O((2s)^t \log n)$ communication and polynomial time and space on s servers.*

4 Low-rank Approximations

For a matrix A , define $f_k(A)$ as:

$$f_k(A) = \min_{X: \text{rank}(X) \leq k} \|A - X\|_F.$$

Recall that the **rank k approximation problem** is the following: Given a $n \times d$ matrix A , and $\varepsilon > 0$, find a $n \times d$ matrix B of rank at most k such that $\|A - B\|_F \leq (1 + \varepsilon) \cdot f_k(A)$.

We will give bounds in terms of d, k, n and ε . An interesting range is when $d, k \in \tilde{O}(1)$ and $n \rightarrow \infty$; our algorithms’ communication is at most polylogarithmic in this range.

4.1 Row-Partition Model

This model is appropriate in situations where data points are rows of a $n \times d$ matrix A . The set of data points is partitioned (arbitrarily) into s servers. Let n_t denote the number of data points stored in server t with $n_1 + n_2 + \dots + n_s = n$. Let $A_t, t = 1, 2, \dots, s$ denote the $n_t \times d$ matrix of data points stored in server t . [Note: each row resides wholly in one server.]

In the Streaming (1-pass) model, the rank- k approximation problem cannot be solved with polylogarithmic space, even when d, k are $\tilde{O}(1)$ and the matrix is presented in row-order as shown by Clarkson and Woodruff [16].

Theorem 4.1 (Theorem 4.10 of [16].) *Suppose a $n \times d$ matrix A is input to a streaming algorithm in row order and assume that $d \in \Omega(k)$. If the algorithm solves the Rank- k Approximation Problem with probability of error at most $1/3$, then it must use $\Omega(nk/\varepsilon)$ bits of space.*

In contrast, we will show that there are nimble algorithms which can solve the problem in polynomial time and polylogarithmic communication when $d, k \in \tilde{O}(1)$. One caveat: Clarkson and Woodruff's lower bounds here as well one presented later require the full rank k approximation matrix to be output and this per se requires $\Omega(nd)$ bits. But in the cloud set-up, we will not require any one server to have the full matrix. Nevertheless, the approximation they together compute must be valid for the whole matrix.

We will describe three algorithms. The first algorithm is very simple:

First Algorithm

- Server t computes $A_t^T A_t$ and communicates it to server 1.
- Server 1 computes $B = \sum_{t=1}^s A_t^T A_t$. It does the SVD of B to find the its top k singular vectors - v_1, v_2, \dots, v_k and communicates these to all the servers.
- Server t computes $C_t = A_t \sum_{i=1}^k v_i v_i^T$.

The proof of correctness is also simple: we have for any vector v :

$$|Av|^2 = \sum_{t=1}^s |A_t v|^2 = v^T \sum_t A_t^T A_t v.$$

So the top k singular vectors of A are just the top k singular vectors of B , namely, v_1, v_2, \dots, v_k . It is well-known that the best rank k approximation to A is $A \sum_{i=1}^k v_i v_i^T$ and so in this case, in fact, we can take $\varepsilon = 0$. This is summarized by:

Theorem 4.2 *Suppose the rows of the input $n \times d$ matrix A are partitioned among s servers with an $n_t \times d$ matrix A_t in server t . There is a nimble algorithm which on termination leaves an $n_t \times d$ matrix C_t in server t such that (denoting C as the $n \times d$ matrix made up from the C_t), C is the best rank k approximation to A . Each server uses polynomial time and the total communication is $O(sd^2)$ real numbers.*

The Second Algorithm

- For $t = 1, 2, \dots, s$, server t (in parallel) does (truncated) SVD of A_t to find a $n_t \times k$ matrix P_t and a $k \times d$ matrix R_t such that $P_t R_t$ is the best rank k approximation to A_t . [The rows of R_t are the right singular vectors of A_t .]
- Server t communicates R_t and $P_t^T P_t$ to server 1.
- Server 1 finds $B = \sum_{t=1}^s R_t^T P_t^T P_t R_t$. It does the SVD of B to find the top k singular vectors v_1, v_2, \dots, v_k of B . It communicates these vectors to all servers.
- Server t computes $C_t = A_t \sum_{i=1}^k v_i v_i^T$.

Theorem 4.3 *The matrix C (comprising of the C_t found by the algorithm) satisfies:*

$$\|A - C\|_F \leq 3f_k(A).$$

The algorithm uses polynomial time and communicates $O(skd)$ real numbers.

Proof. Define an $n \times d$ matrix W :

$$W = \begin{pmatrix} P_1 R_1 \\ P_2 R_2 \\ \dots \\ P_p R_p \end{pmatrix}.$$

We have

$$\|A - W\|_F^2 = \sum_{t=1}^s \|A_t - P_t R_t\|_F^2 = \sum_{t=1}^s f_k(A_t)^2 \leq f_k(A)^2, \quad (2)$$

the last because if A' is the best rank k approximation to the full matrix A and we partition the rows of A' into A'_1, A'_2, \dots, A'_s as for A , then $f_k(A_t) \leq \|A_t - A'_t\|_F$ and $\sum_{t=1}^s \|A_t - A'_t\|_F^2 = \|A - A'\|_F^2$.

Let V be the $d \times k$ matrix whose columns are the top k right singular vectors of W . We will see later a nimble algorithm to compute V . Assume for now V is known. Let A' be the best rank k approximation to A .

$$\begin{aligned} \|A - WVV^T\|_F &\leq \|A - W\|_F + \|W - WVV^T\|_F \leq \|A - W\|_F + \|W - A'\|_F \\ &\leq \|A - W\|_F + \|W - A\|_F + \|A - A'\|_F \leq 3 \cdot f_k(A), \end{aligned} \quad (3)$$

where, for the second inequality, we have used the fact that $\|W - WVV^T\|_F = \min_{X: \text{rank}(X) \leq k} \|W - X\|_F \leq \|W - A'\|_F$ and for the last inequality, we have used (2).

Now to find V nimbly: Let X be a general $d \times k$ matrix whose columns form an orthonormal set of vectors. [As is well known, the X which maximizes $\|WXX^T\|_F$ is V .] We have:

$$\begin{aligned} \|WXX^T\|_F^2 &= \sum_{t=1}^s \|P_t R_t X X^T\|_F^2 = \sum_{t=1}^s \text{Tr} X X^T R_t^T P_t^T P_t R_t X X^T \\ &= \text{Tr} \left[X X^T \left(\sum_{t=1}^s R_t^T P_t^T P_t R_t \right) X X^T \right] = \text{Tr} X X^T B X X^T. \end{aligned}$$

It follows that the algorithm correctly computes V and this finishes the proof of correctness. The communication bound is also simple: Only R_t (each kd reals), $P_t^T P_t$ (each k^2 reals) and v_1, v_2, \dots, v_k (kd reals) are communicated (per server). We may assume without loss of generality that $k \leq d$, since otherwise, we can just keep $C_t = A_t$ and meet the requirements of the theorem. So $k^2 \in O(kd)$ and the theorem is proved. \square

It is an interesting open question to improve the factor of 3 in the theorem, hopefully to $1 + \varepsilon$. **Proof.**(of Thm. 1.4.) To prove this theorem, we have to achieve better communication efficiency for sparse matrices. In this case, we use a result of Boutsidis, Drineas and Mahoney [10] (improving on a line of work [19, 21, 20, 33, 11]) algorithm for row/column-row based relative error low-rank matrix approximation. They have shown that from A_t , we can find $O(k)$ rows of A_t so that in their span, there is an approximation D_t to A_t so that $\|A_t - D_t\|_F \leq 2 \cdot f_k(A_t)$. Now this algorithm will be essentially the same as the second, except, now R_t will be the $O(k)$ rows of A_t found above. We defer the details. Now the communication is $O(sk)$ actual rows of A and $P_t^T P_t$ (which is $O(k^2)$ real numbers), and achieves a factor 5 approximation. This can be reduced to $3 + \varepsilon$, by using $O(k/\varepsilon)$ rows of A_t in place of R_t and getting

$$\|A_t - D_t\|_F \leq (1 + \frac{\varepsilon}{2}) f_k(A_t).$$

Using the same analysis as in the proof of Theorem 1.4, we get the approximation factor of $1 + (\varepsilon/2) + 1 + (\varepsilon/2) + 1 = 3 + \varepsilon$. \square

4.2 Arbitrary Partition

The crucial point of the nimble algorithm for finding a rank k approximation to A with relative error ε in the arbitrary partition model will be a random “projection matrix” P . P will be an $r \times n$ matrix, where, $r \in \tilde{O}(d)$ and we will do computations on PA instead of on A . For this to work, we will need that for all vectors $x \in \mathbf{R}^d$, $|PAx| \approx |Ax|$. Note that of course, it suffices to prove for all unit length vectors y in the d -dimensional space spanned by the columns of A , we have $|Py| \approx 1$.

The entries of P will be random ± 1 . If they are completely independent, we can have a theorem very similar to the Johnson-Lindenstrauss theorem [4, 1, 5], but complete independence has a high space requirement for storage and communication. At the other extreme, the paper of Alon et al [3] uses (in effect) P with mutually independent rows, but only say 4-way independence inside each row to prove for **one vector** v that $|Pv| \approx |v|$. We need here to prove this for exponentially (in d alone) many vector lengths (namely all vectors in an ε net of the column space of A). For this, we will use greater way independence (but not full independence). The proof that the failure probability is exponentially (in d) low for each vector is also a more delicate than usual, since as we point out, the usual Höfding inequality fails. So, we choose here to give a from-first-principles description and rigorous proofs of the non-standard parts.

The projection matrix P will have the following properties:

1. P is $r \times n$, where, r is in $\Omega(d/\varepsilon^2)$
2. Each entry of P is ± 1 .
3. The entries in each row of P are m -way independent, where, $m = \Omega(d \log(2/\varepsilon))$. I.e., for each i , and j_1, j_2, \dots, j_m and $\alpha_1, \alpha_2, \dots, \alpha_m \in \{-1, +1\}$,

$$\text{Prob}(P_{ij_1} = \alpha_1; P_{ij_2} = \alpha_2; \dots; P_{ij_m} = \alpha_m) = 2^{-m}.$$

4. The r rows of P are mutually independent. I.e., for any r vectors $v_1, v_2, \dots, v_r \in \{-1, +1\}^n$,

$$\text{Prob}(P_1 = v_1; P_2 = v_2; \dots; P_r = v_r) = \text{Prob}(P_1 = v_1) \text{Prob}(P_2 = v_2) \dots \text{Prob}(P_r = v_r).$$

Lemma 4.4 *Assume n is a power of 2 and let F be the finite field with n elements, where, each element is represented as a $\log_2 n$ bit string. Let p_1, p_2, \dots, p_r be r polynomials, each of degree $m-1$ whose coefficients are all picked uniformly and mutually independently at random from F . Define P as follows: P_{ij} is the leading bit of the k -bit binary expansion of $p_i(j)$ (polynomial p_i evaluated at j over F .) Then,*

- P has properties (1) through (4) above.
- The p_i can all be communicated with $O(d^2 \log n \log(1/\varepsilon)/\varepsilon^2)$ bits.

The proof of the first part is routine and will be given in the full paper. The second part is obvious.

Now, we prove that for every $x \in \mathbf{R}^d$ simultaneously, we have with high probability that $\|PAx\|$ is an estimate of Ax with relative error ε .

Lemma 4.5

$$\text{Prob} \left(\exists x \in \mathbf{R}^d : \|PAx\|^2 \notin ((1-\varepsilon)r\|Ax\|^2, (1+\varepsilon)r\|Ax\|^2) \right) \leq c_1 e^{-c_2 d}.$$

Proof. The proof starts with an ε net with ε^{-cd} elements for the set of unit length vectors y in the column space of A . This space is d dimensional and so it is clear that such an ε net exists. Now for each y in the net, we wish to prove that the probability that $\|Py\|^2 \notin ((1-\varepsilon)r, (1+\varepsilon)r)$ is at most ε^{-cd} . We point out that standard Höfding inequality will not work, since $|P_i y|$ could be (in the worst-case) as much as \sqrt{n} , since $|P_i| = \sqrt{n}$ and $|y| = 1$. Inequalities exploiting finite (rather than exponential) moments of $|P_i y|$ are necessary. To this end, we note that for any $p \leq m$, p even, we have

$$\mathbb{E}((P_i \cdot y)^p) = \sum_{j_1, j_2, \dots, j_p} y_{j_1} y_{j_2} \cdots y_{j_p} \mathbb{E}(P_{i_{j_1}} P_{i_{j_2}} \cdots P_{i_{j_p}}),$$

where, j_1, j_2, \dots, j_p are any p indices (not necessarily distinct) from $\{1, 2, \dots, n\}$. Terms where any $j \in \{1, 2, \dots, n\}$ occurs an odd number of times are zero using m -way independence. Also P_{ij} to even power is 1. So it is easy to see that

$$\mathbb{E}((P_i \cdot y)^p) = \sum_{(j_1, j_2, \dots, j_\ell), (d_1, d_2, \dots, d_\ell)} \binom{p}{d_1, d_2, \dots, d_\ell} y_{j_1}^{d_1} y_{j_2}^{d_2} \cdots y_{j_\ell}^{d_\ell},$$

where, the d 's are even positive integers summing to p . Note that we have

$$\binom{p}{d_1, d_2, \dots, d_\ell} \leq p^{p/2} \binom{(p/2)}{(d_1/2), (d_2/2), \dots, (d_\ell/2)}.$$

So we get

$$\mathbb{E}((P_i \cdot y)^p) \leq p^{p/2} \left(\sum_{j=1}^n y_j^2 \right)^{p/2} = p^{p/2}.$$

Put $X_i = (P_i \cdot y)^2 - \mathbb{E}((P_i \cdot y)^2)$ for $i = 1, 2, \dots, r$. Then $\mathbb{E}X_i = 0$ and $\text{Var}(X_i) = 1$. By the above,

$$\mathbb{E}(X_i)^p \leq 2^p (\mathbb{E}((P_i \cdot y)^{2p}) + 2^p \leq (cp)^p$$

for a constant c . Now taking m to be even and applying Theorem 1 from [28], we get:

$$\mathbb{E}(|\|Py\|^2 - r|^m) = \mathbb{E}((\sum_{i=1}^r X_i)^m) \leq (crm)^{m/2}.$$

Using Markov's inequality, we therefore get

$$\text{Prob}(|\|Py\|^2 - r| \geq \varepsilon r) \leq \frac{\mathbb{E}(|\|Py\|^2 - r|^m)}{(\varepsilon r)^m} \leq \frac{(crm)^{m/2}}{(\varepsilon r)^m} = \left(\frac{cm}{\varepsilon^2 r} \right)^{m/2} \leq \varepsilon^{-c_4 d}$$

for our choice of $r = Cm/\varepsilon^2$ and $m = Cd \log(2/\varepsilon)$.

Now since there are only $(2/\varepsilon)^d$ elements in the ε -net, this holds simultaneously for all y in the net.

Suppose now z is any unit vector (not necessarily in the net). Then, by repeated approximation, we can express $z = y_1 + \alpha_2 y_2 + \alpha_3 y_3 + \cdots$, where, y_1, y_2, \dots are unit vectors in the ε net and $\alpha_k \leq \varepsilon^{k-1}$. From this, the Lemma follows in a standard fashion. \square

We now state the complete algorithm.

- Server 1 generates the coefficients of the polynomials p_1, p_2, \dots, p_r as described in Lemma (4.4). It communicates the coefficients of these polynomials to all servers.
- Server t uses the polynomials to compute the matrix P as in Lemma (4.4). It computes PA_t and communicates it to server 1.
- Server 1 computes $\sum_{t=1}^s PA_t = PA$. It does SVD on PA to find the top k singular vectors v_1, v_2, \dots, v_k and communicates them to all servers.
- Server t computes $C_t = A_t \sum_{i=1}^k v_i v_i^T$.

We are now ready for the main theorem for an arbitrary partition.

Proof. (of Theorem 1.5.) Form an orthonormal basis of \mathbf{R}^d using the right singular vectors of PA . Let v_1, v_2, \dots, v_d be the basis.

$$\begin{aligned} \|A - A \sum_{i=1}^k v_i v_i^T\|_F^2 &= \sum_{i=k+1}^d |Av_i|^2 \leq (1 + \varepsilon)^2 \sum_{i=k+1}^d |PAv_i|^2 \\ &= (1 + \varepsilon)^2 f_k^2(PA). \end{aligned}$$

Also, suppose now u_1, u_2, \dots, u_d is an orthonormal basis consisting of the singular vectors of A . Then, we have

$$\begin{aligned} f_k(PA)^2 &\leq \|PA - PA \sum_{i=1}^k u_i u_i^T\|_F^2 = \sum_{i=k+1}^d |PAu_i|^2 \\ &\leq (1 + \varepsilon)^2 \sum_{i=k+1}^d |Au_i|^2 = (1 + \varepsilon)^2 f_k(A)^2. \end{aligned}$$

Thus,

$$\|A - A \sum_{i=1}^k v_i v_i^T\|_F^2 \leq (1 + \varepsilon)^4 f_K(A)^2$$

and the theorem follows. \square

In contrast, in the Streaming model, even with multiple passes, the problem cannot be solved with polylog space.

Theorem 4.6 (Theorem 4.14 of [16]) *For any $k, 1 \leq k \leq \min\{n, d\}$ and any $\varepsilon > 0$, any multi-pass algorithm for the Rank k approximation problem for input presented in arbitrary order which has probability of error at most $1/3$ must use $\Omega((n + d)k \log(nd))$ bits of space.*

5 Clustering

In this section, we prove Theorem 1.6. Resource (time and communication) bounds will be given in terms of n, d, k , which can take on any values. Similar to low-rank approximation, our main interest here is when $n \gg d$, with $n, d \rightarrow \infty$ but $k \in O(1)$.

First, a note of caution. For data points generated from a spherical Gaussian of standard deviation 1 in each direction, data points are at distance $O(\sqrt{d} \pm 1)$ from the mean. So, using data points as cluster centers (as in say the well-known k -means ++ heuristic [6]) will not be a valid approximation. Also, a $(1 + \varepsilon)$ approximation to the optimal k -means clustering say (or to the optimal k -median clustering) does not guarantee a valid approximation to C . A simple example of this is when data points are generated by a mixture of two spherical Gaussians in \mathbf{R}^d with variance 1 in every direction and their means separated by $\Omega(1)$. It is easy to see that the clustering C dividing the data points according to which Gaussian they were generated from is proper. But the mean-squared distance of a data point to a cluster center is d and so the near optimal clustering could have an error of εnd . So this does not rule out the cluster centers found from being $\Omega(\sqrt{\varepsilon d})$ away from the centers in C ! In particular, this means that performing a random projection up front that preserves all pairwise distances to within relative error $(1 + \varepsilon)$ does not suffice.

In contrast to the remark above that doing near-optimal clustering in \mathbf{R}^d does not necessarily yield a valid approximation, it is known that a constant factor approximation to the optimal k -means clustering in the projection to the k -dimensional SVD sub-space does give us a valid approximation:

Lemma 5.1 (*Claim 3.3 of [29] and Lemma 5.1 of [30]*) *Let V be the space spanned by the top k (right) singular vectors of A . A constant factor approximation to the optimal k -means clustering of the rows of A projected to V is a valid approximation to C .*

The nimble algorithm for clustering the projected points will also crucially use the important result of Chen [15] on “core-sets”. Chen gave the first construction of a coreset of size polynomially bounded in dimension.

Theorem 5.2 [15] *For any set W of n points in \mathbb{R}^d space to be clustered into k clusters, in polynomial time, we can find a weighted subset (called a coreset) X of $\tilde{O}(dk^2)$ points among them such that for any set Y of k centers, the cost of clustering W with centers Y is within a constant factor of the cost of clustering X with Y as centers (according to the k -means objective).*

Using the above two properties, and the low-rank approximation of the previous section, we will prove that clustering can be achieved by a nimble algorithm.

Proof. (of Theorem 1.6.) The algorithm will project the points to their k -dimensional SVD subspace V , then cluster the projected points in the SVD subspace. It follows from Lemma 5.1 that this will give a valid clustering. What remains is to make the two parts — finding V and clustering the projected points — nimble. The first is already done by the first algorithm of the last section. At the end of that algorithm, each server has the top k right singular vectors of the whole matrix A and so can do the projection of its data points to V . But we still cannot communicate the n projected points to figure out a near optimal clustering.

We now apply Chen’s result for the points in the SVD subspace V , so the coreset X given by his theorem has size $\tilde{O}(k^3)$ only. So one could k -cluster X instead of the full point set W . Further, Chen’s algorithm can be made nimble provided each server already has a constant factor approximation to the optimal k -clustering of W - namely, provided each server has the same set Y' of $O(k)$ centers so that the cost of clustering W with Y' as centers is within a constant factor of the optimal k -means cost for W . We describe briefly Chen’s algorithm and how it can be made nimble. The algorithm partitions W into W_1, W_2, \dots , where, W_i is the set of points in W with the

i th point of Y' (call this y_i) as its closest point in Y' . Then, we partition W_i further into “rings”, namely, we let

$$W_{ij} = \{x \in W_i : |x - y_i| \in (R \cdot 2^{j-1}, R \cdot 2^j)\},$$

for a suitable R . Then, the algorithm picks uniformly at random a certain number of points in each W_{ij} and together these form the coreset. Now to make this nimble, with Y' on hand, each server can find its part of W_{ij} and communicate only the cardinality of it. Based on the cardinality, it will randomly draw a certain number of points from its part of W_{ij} to be included in the coreset and communicate this to all the servers at the end of which all servers have the coreset.

It now remains only to see that we can nimbly compute a set Y' as above. But this is straightforward — each server just finds a constant factor approximation to the optimal k clustering of its own data points and communicates these centers. Y' will simply be the union of all these centers. To summarize, here is the algorithm.

Clustering Algorithm

- The servers first use the first algorithm of the last section (communicating $A_t^T A_t$ to server 1 which then finds the top k singular vectors of $\sum_{t=1}^s A_t^T A_t$ which are also the top k singular vectors of A and communicates them to all servers) to find the top k singular vectors v_1, v_2, \dots, v_k of A . Let V = the span of v_1, v_2, \dots, v_k .
- Server t projects the rows of A_t onto V .
- Server t finds a factor 2 approximation to the optimal k -means clustering of its **projected points**.
- Server t broadcasts the k centers found in the last step to all servers. So now all servers know the set Y' of sk centers found; let $Y' = \{a_1, a_2, \dots, a_{sk}\}$.
- Use the algorithm of Chen as described above.

□

6 Conclusion

We have presented algorithms and analysis for frequency moments, counting homomorphisms, low-rank approximation and clustering, in that order. The model and results raise several interesting questions: (1) Privacy. In the distributed setting, it is often the case that a computation must be done privately [22, 2]. Which problems have *privacy-preserving* nimble algorithms? (2) Graph problems. Do basic graph problems such as finding shortest paths or finding large matchings have nimble algorithms? (3) What is the class of homomorphisms that can be approximately counted with nimble algorithms? (4) Can we prove lower bounds on sampling from natural distributions with nimble algorithms? (e.g., sample item according to its k 'th frequency moment).

In a practical cloud set-up, there might be some control over where each piece of data is stored. However, allocating optimally to achieve best “run-time” efficiency is clearly very hard. We assumed that the partition of data into processors is adversarial here, but there is another extreme possibility: namely, each piece of input data is assigned on arrival to a uniformly randomly picked server. This random partition model may allow more problems to be solved by nimble algorithms.

The communication complexity of our rank- k approximation algorithm is $\tilde{O}(sd^2/\varepsilon^2)$. David Woodruff showed us that using a 2-round algorithm, the first term can be improved to $\tilde{O}(skd/\varepsilon^2)$. Jelani Nelson observed that we only need $O(\log d)$ -wise independence (rather than $O(d)$ -wise independence) reducing the number of bits to communicate the projection matrix to $\tilde{O}(1/\varepsilon^2)$.

Acknowledgements. We are grateful to Dick Karp, David Woodruff and Jelani Nelson for helpful discussions.

References

- [1] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003.
- [2] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, SIGMOD ’00, pages 439–450, New York, NY, USA, 2000. ACM.
- [3] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.
- [4] Rosa I. Arriaga and Santosh Vempala. An algorithmic theory of learning: Robust concepts and random projection. In *FOCS*, pages 616–623, 1999.
- [5] Rosa I. Arriaga and Santosh Vempala. An algorithmic theory of learning: Robust concepts and random projection. *Machine Learning*, 63(2):161–182, 2006.
- [6] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA*, pages 1027–1035, 2007.
- [7] Maria-Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. Distributed learning, communication complexity and privacy. In *COLT*, pages 26.1–26.22, 2012.
- [8] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004.
- [9] R. Bekkerman, M. Bilenko, and J. Langford. *Scaling up machine learning: Parallel and distributed approaches*. Cambridge Univ Pr, 2011.
- [10] Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. Near-optimal column-based matrix reconstruction. *CoRR*, abs/1103.0995, 2011.
- [11] Christos Boutsidis, Petros Drineas, and Michael Mahoney. An improved approximation algorithm for the column subset selection problem. In *SODA*, pages 968–977, 2009.
- [12] Amit Chakrabarti. *Data Stream Algorithms*. Lecture notes, Dartmouth College, 2010.
- [13] Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *IEEE Conference on Computational Complexity*, pages 107–117, 2003.

- [14] Moses Charikar, Sudipto Guha, va Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem. In *Proc. of the 31st Annual ACM Symposium on Theory of Computing*, pages 1–10, 1999.
- [15] Ke Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM J. Comput.*, 39(3):923–947, 2009.
- [16] Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. In *STOC*, pages 205–214, 2009.
- [17] Amin Coja-oghlan. Graph partitioning via adaptive spectral techniques. *Comb. Probab. Comput.*, 19(2):227–284, March 2010.
- [18] Graham Cormode, S. Muthukrishnan, and Ke Yi. Algorithms for distributed functional monitoring. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms, SODA '08*, pages 1076–1085, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.
- [19] Amit Deshpande, Luis Rademacher, Santosh Vempala, and Grant Wang. Matrix approximation and projective clustering via volume sampling. *Theory of Computing*, 2(1):225–247, 2006.
- [20] Amit Deshpande and Santosh Vempala. Adaptive sampling and fast low-rank matrix approximation. In *APPROX-RANDOM*, pages 292–303, 2006.
- [21] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-based methods. In *APPROX-RANDOM*, pages 316–326, 2006.
- [22] Cynthia Dwork. The promise of differential privacy: A tutorial on algorithmic techniques. In *FOCS*, pages 1–2, 2011.
- [23] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Applied Statistics*, 28(1):100–108, 1979.
- [24] Hal Daumé III, Jeff M. Phillips, Avishek Saha, and Suresh Venkatasubramanian. Efficient protocols for distributed classification and optimization. In *ALT*, pages 154–168, 2012.
- [25] Hal Daumé III, Jeff M. Phillips, Avishek Saha, and Suresh Venkatasubramanian. Efficient protocols for distributed classification and optimization. *CoRR*, abs/1204.3523, 2012.
- [26] Hal Daumé III, Jeff M. Phillips, Avishek Saha, and Suresh Venkatasubramanian. Protocols for learning classifiers on distributed data. In *AISTATS*, pages 282–290, 2012.
- [27] Piotr Indyk and David P. Woodruff. Optimal approximations of the frequency moments of data streams. In *STOC*, pages 202–208, 2005.
- [28] Ravindran Kannan. A new probability inequality using typical moments and concentration results. In *FOCS*, pages 211–220, 2009.

- [29] Ravindran Kannan and Santosh Vempala. Spectral algorithms. *Foundations and Trends in Theoretical Computer Science*, 4(3-4):157–288, 2009.
- [30] Amit Kumar and Ravindran Kannan. Clustering with spectral norm and the k-means algorithm. *CoRR*, abs/1004.1823, 2010.
- [31] S Muthukrishnan. *Data streams: Algorithms and applications*. Now Publishers Inc, 2005.
- [32] Jeff M. Phillips, Elad Verbin, and Qin Zhang. Lower bounds for number-in-hand multiparty communication complexity, made easy. In *SODA*, pages 486–501, 2012.
- [33] Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In *FOCS*, pages 143–152, 2006.
- [34] David P. Woodruff and Qin Zhang. Tight bounds for distributed functional monitoring. In *STOC*, pages 941–960, 2012.